Intro to Data Structures

Lecture #14 – Interfaces
October 29, 2014
Mark Stehlik

Outline for Today

- HW 5 out Thursday, due 11/3
- HW4 issues:
  - any MyLinkedList has Nodes (and a first/length)
  - treat issues with first separately!
  - But <u>don't</u> use first to traverse the list
- Quiz on Thursday (what type is this, box-and-pointer diagrams, a HW method)
- Java Interfaces

Interfaces

- An interface is a specification of the *way* something works without regard to *how* it is implemented
- Real-life interfaces that you have seen implemented:
  - light switch
  - key/door access
  - car (steering / brake & gas)
- Java interface
  - the spec ([default public/abstract] method declarations) of what the abstract object *has* to do
- A class implements an interface (can implement many)
  - must provide code to implement all the methods specified by the interface (can have additional methods)

An example

- Let's play rock-paper-scissors (to the code!)
- So this works fine, but what if I want a ScissorPlayer instead of a RockPlayer, or two RandomPlayers?
- I could make the parameters to the play method be Objects, but…
  - Objects don't have a move method that returns a String!

Example (continued)

- What I need is a way to specify a generic Player that has a move method that returns a String…
- What I need is a Player *interface*
- Back to the code…

Rock, Paper, Scissors (extended)

- The rules of rock-paper-scissors-lizard-Spock:
  - Scissors cut paper
  - Paper covers rock
  - Rock crushes lizard
  - Lizard poisons Spock
  - Spock smashes scissors
  - Scissors decapitate lizard
  - Lizard eats paper
  - Paper disproves Spock
  - Spock vaporizes rock
  - Rock crushes scissors

Another example

- I've got a Point class (what’s the point?)
- I want to create an array of random points (to the code)
- Now, I want to sort that array of Points. In order to use Arrays.sort, what has to be true? To the API (why, what am I going to look at?)…
- the Points must be *Comparable* (to the API…)

Another example (continued)

- n.b. if a class implements *Comparable*, it should also override the equals method to return true in the same way compareTo returns 0.
- There's one more method consider – overriding hashCode() – but that's for another lecture…

Javadoc (after the quiz on Thursday)

- Comment your code with appropriate tags; see
  http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#tag
- Run javadoc (e.g., `javadoc PointXY.java`)
- If you did it all correctly, you should have an index.html file that describes the API of your class (just like the Java API)
- (you will also have a lot of other html files as well!)